View metadata, citation and similar papers at core.ac.uk
brought to you by CORE
provided by Digital library of Brno University of Technology

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV INTELIGENTNÍCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF INTELLIGENT SYSTEMS

# AUTOMATICKÁ ORGANIZACE DOKUMENTŮ V SOUBOROVÉM SYSTÉMU

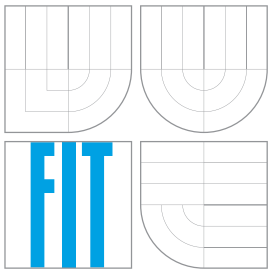
BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE
AUTHOR

RADIM SVÁČEK

BRNO 2015

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV INTELIGENTNÍCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF INTELLIGENT SYSTEMS

# AUTOMATICKÁ ORGANIZACE DOKUMENTŮ V SOUBOROVÉM SYSTÉMU

AUTOMATED DOCUMENT CATEGORIZATION

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE
AUTHOR

RADIM SVÁČEK

VEDOUCÍ PRÁCE
SUPERVISOR

Dr. Ing. PETR PERINGER

BRNO 2015

# Abstrakt

Cílem této bakalářské práce je návrh a implementace inteligentního organizátoru dokumentů v souborovém systému. Práce obsahuje přehled již existujících řešení a popis metod klasifikace dokumentů. Součástí je objektově orientovaný návrh aplikace a základních zásuvných modulů. Program je implementován v jazyce C++ s možností rozšiřování další funkčnosti pomocí nových zásuvných modulů. Výsledná aplikace si klade za cíl ulehčit uživateli práci s netříděnými soubory díky jejich organizaci do adresářové struktury na základě informací získaných z těchto dokumentů, včetně jejich metadat.



# Abstract

This thesis deals with design and implementation of automated document categorization application. Text contains analysis of existing systems and description of classification methods. Thesis includes object oriented design of application including plugins. Application is implemented in C++ with possibility of functionality extension through plugins. Application aims to simplify work with unsorted documents through their organization into directory structure based on informations extracted from files, including their metadata.



# Klíčová slova

organizace dokumentů, klasifikace souborů, objektově orientovaný návrh, C++, Qt, JPEG, EXIF, PDF, HTML, zásuvné moduly

# Keywords

document organization, file classification, object oriented design, C++, Qt, JPEG, EXIF, PDF, HTML, plugins


# Citace

Radim Sváček: Automatická organizace dokumentů
v souborovém systému, bakalářská práce, Brno, FIT VUT v Brně, 2015

# Automatická organizace dokumentů v souborovém systému

## Prohlášení

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně pod vedením pana Dr. Ing. Petra Peringera. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

.....
Radim Sváček
18. května 2015


## Poděkování

Na tomto místě bych velice rád poděkoval vedoucímu této práce, Dr.Ing. Petru Peringerovi, za cenné rady, ochotu a čas strávený při konzultacích.



© Radim Sváček, 2015.
*Tato práce vznikla jako školní dílo na Vysokém učení technickém v Brně, Fakultě informačních technologií. Práce je chráněna autorským zákonem a její užití bez udělení oprávnění autorem je nezákonné, s výjimkou zákonem definovaných případů.*

# Obsah

# Kapitola 1

# Úvod

V dnešní době používá počítač stále více uživatelů pro správu svých dokumentů, ať už jde o fotografie, knihy v elektronické podobě, filmy či další multimediální soubory. Ovšem málokterý uživatel si své dokumenty spravuje tak, aby o nich měl neustále přehled, přičemž počet uložených souborů neustále narůstá. Často pak má ve svém počítači několik složek s neuspořádanými soubory, kde poté jen stěží hledá požadovaný soubor. Jedním z možných řešení tohoto problému je přehledná adresářová struktura, ve které jsou dokumenty ukládány na základě jejich vlastností. Toho lze dosáhnout pravidelným ručním přesunem souborů, jenž je ovšem časově náročný. Lepší možností je využít některý z nástrojů k tomu určených, které dokáží soubory uspořádat. Cílem této práce je návrh a implementace aplikace umožňující inteligentní organizaci dokumentů do adresářové struktury na základě požadavků uživatele.

Text práce je rozdělen do čtyř kapitol. Úvodní kapitola seznamuje čtenáře s vybranými metodami klasifikace dokumentů. Dále je zde zmíněno několik již existujících řešení problému organizace velkého množství souborů podle jejich vlastností. Druhá kapitola popisuje návrh aplikace s důrazem na rozšiřitelnost pomocí zásuvných modulů. Základem je automatická klasifikace souborů různých typů a jejich organizace do adresářové struktury na základě požadavků uživatele. Dále je zde popsáno uživatelské rozhraní pro interakci s uživatelem. Ve třetí kapitole je popsána samotná implementace jádra aplikace a základních zásuvných modulů pro klasifikaci dokumentů podle návrhu. Také jsou zde zmíněny knihovny použité při vývoji a postup testování. Na závěr je shrnuta dosavadní práce včetně jejího zhodnocení. Dále je zde zmíněno i další možné implementační rozšíření.

# Kapitola 2

# Analýza současného stavu

Tato kapitola se zaměřuje na analýzu problému organizace dokumentů a je rozdělena do tří částí. V úvodu je stručně popsána aplikace, jejíž návrh a implementace je cílem této práce. Následující část je věnováno rozboru již existujících řešení tohoto problému, či problémů podobných, tedy implementovaných aplikací dostupných běžnému uživateli. Další část této kapitoly se pak věnuje klasifikaci dokumentů a analýze vybraných nejznámějších metod klasifikace.

## 2.1 Vlastnosti aplikace typu organizátor dokumentů

Dříve se k účelům organizace dokumentů téměř výhradně využívaly jednoduché aplikace typu správce souborů. Mezi nejznámější patří například Norton Commander[1] či Midnight Commander[2]. Postupem času ale začaly být i tyto nástroje pro organizaci dokumentů neefektivní. Proto začaly vznikat specializované aplikace, které dokázaly zpracovávat určitý formát souborů, jako například organizátory fotografií či hudebních souborů. Ke komplexním systémům, jež dokáží organizovat různorodé dokumenty, tak scházel již jen krůček.

Cílem aplikace pro organizaci dokumentů je bezpochyby šetřit čas uživatele. Aby se dala označit za inteligentní, musí taková aplikace splňovat určité kritéria. Stěžejní funkcí je přesun, kopírování, mazání či jiná operace se soubory různých typů. Ovšem nejdůležitější části je právě výběr souborů, které se budou zpracovávat. Pravidla, určující, které soubory zpracovat, jsou vždy vytvořeny uživatelem, přičemž mohou obsahovat nepřeberné množství možností, podle čeho soubory vybírat, jako například tyto:

- Název — řetězec neobsahující cestu k souboru, jehož součástí je i přípona
- Formát — určuje typ dokumentu (dle hlavičky či obsahu souboru)
- Obsah — informační hodnota dokumentu (text u souboru PDF či obraz u JPEG)
- Velikost — číslo v bajtech či jeho násobcích (kB, MB, GB)
- Časové údaje — datum/čas manipulace se souborem (vytvoření, změny, přístupu, . . . )
- Specifická metadata — (např. EXIF u formátu JPEG, ID3 u formátu MP3)

[1] www.softpanorama.org/OFM/Paradigm/Ch03/norton_commander.shtml
[2] www.midnight-commander.org/

Na základě těchto vlastností souborů aplikace vybere ty, které podmínkám vyhovují, a dále s nimi pracuje. Další operace jsou opět určeny uživatelem. Inteligentní organizátor by měl umět roztřídit tyto soubory do adresářové struktury dle zvolených pravidel. Příkladem může být velké množství fotografií v jediném adresáři. Aplikace může například tyto soubory roztřídit do složek podle data pořízení, přičemž vybere jen ty, které byly vyfoceny přesně specifikovaným fotoaparátem.

## 2.2 Analýza existujících organizátorů dokumentů

Vzhledem k požadavkům uvedeným v předchozí části, jež aplikace typu organizátor dokumentů musí splňovat, byla vybrána existující řešení pro bližší analýzu. Pro přehlednost byly vybrané aplikace rozděleny do sekcí komerčních a volně šiřitelných/freeware řešení. Každé řešení je analyzováno z hlediska nabízených funkcí, zpracovávaných formátů dokumentů a také uživatelského rozhraní. Z analýzy existujících řešení je možno vycházet i při návrhu implementace, kdy je vhodné se inspirovat kvalitními částmi jednotlivých řešení a zároveň se snažit vyvarovat chyb, které ona řešení obsahují.

### Komerční organizátory dokumentů

Jde o zpoplatněné aplikace, často dostupné v různých verzích, které podle ceny zpřístupňují více funkcí. Většina těchto aplikací nabízí možnost vyzkoušení, a to jako trial verzi, tedy aplikaci obsahující plnou funkčnost aplikace po určitou dobu, či verzi s omezenou možností použití, jako tzv. demo.

**RoboBasket**

Jeden z velmi účinných nástrojů pro organizaci velkého množství souborů je RoboBasket[3]. Aplikace je zpoplatněna a dostupná pouze pro operační systémy Microsoft Windows, k dispozici je také trial verze platná na 15 dní.

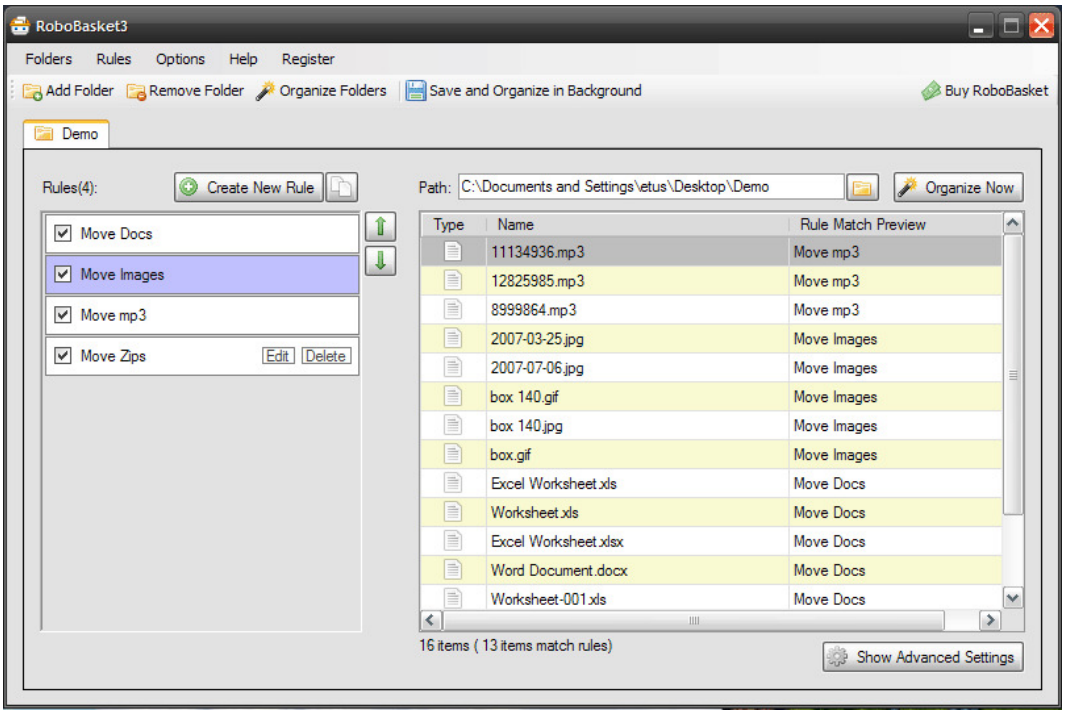
Tento nástroj umí roztřídit soubory na základě jejich jména, typu, velikosti, data apod. Zároveň u souborů MP3 dokáže zpracovat tagy označující interpreta či album, stejně jako u fotografií dokáže pracovat s metadaty ve formátu Exif. Soubory je možné třídit do podadresářů, jež jsou vytvářeny na základě zmíněných vlastností tříděných dokumentů. Další zajímavou funkcí aplikace je možnost běhu na pozadí, kdy jsou sledovány vybrané složky. V případě, kdy je do některé ze sledovaných složek přidán nový soubor, aplikace zjistí, zdali vyhovuje vytvořeným pravidlům a případně vykoná vybranou operaci. Další možností, jak pravidelně organizovat své soubory, je periodické provádění vytvořených pravidel nad vybranou složkou, a to v uživatelem stanoveném časovém rozmezí.

Uživatelské rozhraní aplikace je velmi intuitivní, jak lze vidět na snímku aplikace na obrázku 2.1. Hlavní okno aplikace obsahuje přehled vytvořených pravidel a také zobrazuje obsah aktuálně vybrané složky určené k organizaci. I vytváření nových pravidel pro organizaci je velmi jednoduché, kdy je možné pomocí techniky Drag&Drop vybrat požadované vlastnosti souborů.

**DiskSorter**

Velmi komplexním systémem je DiskSorter[4]. Samotná aplikace je dostupná ve čtyřech

[3] www.robobasket.com
[4] www.disksorter.com




Obrázek 2.1: Snímek aplikace RoboBasket

placených verzích, přičemž všechny jsou spustitelné pouze na operačních systémech Microsoft Windows.

Umožňuje analýzu a klasifikaci více jak 3000 formátu souborů. Výsledkem klasifikace je přehledný souhrn obsahující informace o jednotlivých typech souborů. Ty jsou rozděleny do jednotlivých kategorií, o kterých je také zobrazeno mnoho informací. Ovšem krom toho umožňuje DiskSorter i soubory na disku přesouvat. Díky kvalitní klasifikaci je možno soubory přesouvat či kopírovat na základě různých pravidel zvolených uživatelem. Nejedná se ovšem o jediné funkce celé aplikace, která je mnohem rozsáhlejší a robustnější. Tomu je přizpůsobeno i grafické rozhraní, které se může zdát na první pohled nepřehledné a složité.

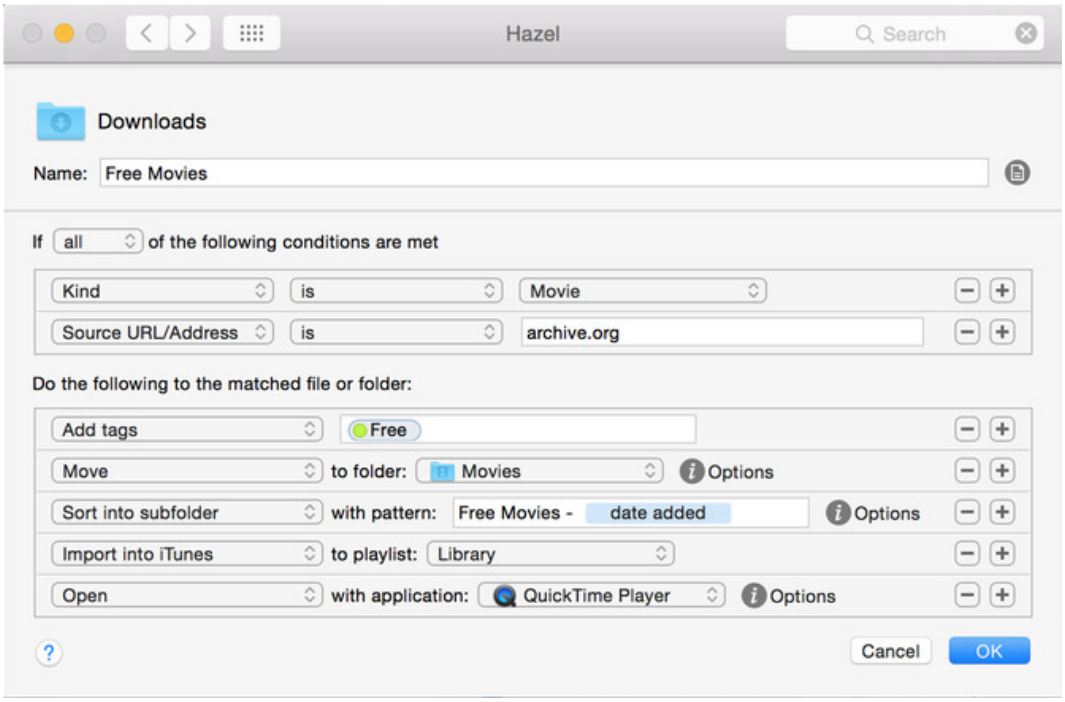
**Hazel**

Jedno z nejpropracovanějších řešení je aplikace Hazel[5]. Nevýhodou této aplikace je fakt, že je dostupná pouze pro Mac OS. Ačkoliv jde o placený produkt, nabízí možnost dvoutýdenní trial verze.

Aplikace umožňuje organizaci zdrojového adresáře na základě předvytvořených pravidel. Tyto pravidla může uživatel vytvářet pomocí jednoduchého nástroje, kdy může vybrat jak podmínky, které musí vybraný soubor splňovat, aby byl zpracován, tak i následné operace se souborem. Podmínek pro výběr souboru je možno vybrat více a spojovat je pomocí logických operátorů. Samotné operace se soubory pak umožňují jejich přesun či kopírování, a to i do podadresářů na základě vybraných pravidel.

Velkou výhodou této aplikace je grafické uživatelské prostředí, které je velmi intuitivní a propracované, jak lze vidět na obrázku 2.2. Ačkoliv Hazel nabízí opravdu širokou

[5] www.noodlesoft.com/hazel.php




Obrázek 2.2: Snímek aplikace Hazel při vytváření pravidel

škálu možností práce se soubory, jsou tyto schopnosti aplikace uživateli nabízeny ve srozumitelné podobě, čímž se stává velmi mocným nástrojem. Další výhodou je pak možnost pracovat i s iTunes či iPhoto.

### Advanced file organizer

Zaměřením programu Advanced file organizer[6] je spíše indexace souborů a uložení podstatných informací do vlastní databáze. Díky tomu není nutné soubory vůbec přesouvat, ale je možné je ponechat ve stávajícím umístění. Díky účinnému vyhledávacímu enginu je pak nalezení požadovaných souborů jednoduchou záležitostí, jelikož aplikace zpracovává krom názvu a typu i metadata souborů.

Jako zdroj souborů přitom nemusí nutně sloužit pouze pevný disk počítače, ale například i síťový disk, CD či DVD. Další výhodou je export informací do formátu tabulkového procesoru.

## Freeware a opensource organizátory dokumentů

Jedná se buďto o aplikace vyvíjené jako open source, tedy software s otevřeným kódem, které jsou nejčastěji šířené pod licencí GNU GPL. Druhou variantou jsou aplikace šířené zdarma pouze v binární podobě, tedy tzv. freeware.

### DropIt

Velmi silný nástroj pro organizaci dokumentů je DropIt[7]. Aplikace DropIt je volně šiřitelná pod licencí GNU GPL, vytvořena je pro OS Microsoft Windows.

[6] www.softprime.com
[7] www.dropitproject.com

Tento nástroj umožňuje provádět dvacet různých operací s vybranými soubory. Kromě klasického kopírování, přesunutí či mazání souborů, dokáže také například soubory komprimovat, spojovat, nahrát na server, poslat emailem či vytvořit jejich katalog. Soubory dokáže třídit do adresářové struktury, podadresáře dokáže vytvářet podle vlastností souborů.

Ačkoliv výběr souborů, které se budou organizovat, je možné provést pouze na základě jejich jména a typu, samotné roztřídění je možné přizpůsobit do velké míry podle představ uživatele. Pomocí zápisů relativních cest cílových adresářů se dají soubory roztřídit podle mnoha vlastností, jako například data, velikosti či typu. Více možností pak nabízí pro soubory obsahující více metadat, konkrétně pro hudební soubory ve formátu MP3 a fotografie. Také periodicky kontroluje uživatelem vybrané adresáře, nad kterými provádí uložené operace ihned po nastalé změně v adresáři, např. po přidání nového souboru do složky. Veškerou svou práci pak zapisuje do logů.

Grafické uživatelské rozhraní může být pro nezkušeného uživatele poněkud složitější, jelikož je přespříliš minimalistické. Zejména pak výběr cílové adresy souborů je nutné zapsat pomocí relativní cesty. Přitom není k dispozici žádný nástroj, jenž by uživateli dokázal ulehčit tento zápis, pouze strohý výpis všech dostupných vlastností, podle kterých lze relativní cestu vytvořit.

**Limagito FileMover**

Dalším analyzovaným nástrojem je utilita Limagito FileMover[8], kterou je možno získat v binární podobě jako freeware. Aplikaci je možno používat pouze na OS Microsoft Windows, přičemž zdarma je pouze základní verze programu, která ovšem disponuje mnoha funkcemi. Omezení se týká především možnosti použití zdroje souborů z FTP serveru apod.

Dokáže manipulovat se soubory pomocí uživatelských pravidel, přičemž může soubory přesouvat, kopírovat či mazat. Pravidla uživatel zadává pomocí přehledného průvodce, kdy je možné nastavit podmínky určující, které soubory se budou zpracovávat. Na výběr je možnost určení podle názvu, data a velikosti souboru. Zároveň při práci program zapisuje logy a současně probíhá i záloha souborů pro případ nepředvídatelných problémů. Zajímavou funkcí je možnost výběru zdrojové složky. Kromě klasického lokálního adresáře je možno zpracovávat soubory na FTP serveru či pomocí emailových protokolů POP3 a IMAP.

**Digital Janitor**

Aplikace Digital Janitor[9] je šířena jako freeware, přičemž je dostupná pro operační systémy Microsoft Windows XP a novější s požadavkem .NET frameworku.

Poskytuje základní možnosti třídění souborů, a to podle jejich názvu, typu a velikosti. Na základě těchto vlastností může uživatel vytvářet pravidla, pomocí kterých budou soubory ze zdrojové složky přesouvány do vybraných adresářů. Poněkud rozsáhlejší možnosti nabízí při organizaci hudebních souborů, a to nejen ve formátu MP3, které pak dokáže třídit i podle interpreta a alba do složek. Další zajímavou funkcí je pak plánovač operací, kdy lze přesně nastavit časy, kdy budou vybrané procesy probíhat. Digital Janitor poskytuje jednoduché a velmi přehledné GUI pro výběr pravidel pro organizaci.

[8] www.limagito.com
[9] www.davidevitelaru.com/software/digital-janitor/

**Belvedere Automated File Manager**

Další analyzovanou aplikací je pod GNU licencí volně šířená aplikace Belvedere Automated File Manager[10]. Jedná se, jak sám autor píše na oficiálních stránkách aplikace, o napodobení programu Hazel, který je pouze pro systém Mac. Belvedere Automated File Manager je naopak přístupný pouze pro systémy Microsoft Windows.

Aplikace dokáže přesouvat soubory na základě uživatelem zadaných pravidel. Ty mohou obsahovat jméno, typ souboru, datum vytvoření, modifikace a posledního otevření, velikost či obsah souboru a jednotlivé operace jako porovnání apod. Takto vybrané soubory pak aplikace přesune do předem vybraného adresáře.

**Miranda**

Dalším klonem aplikace Hazel je program Miranda[11], jež je šířen pod Apache licencí. Funkčnost aplikace je podobná, hlavním rozdílem je dostupnost na Unixových systémech. Pomocí nástroje Miranda je tak možné přesouvat, kopírovat, přejmenovávat či mazat vybrané soubory na základě vytvořených pravidel.

**File fisher**

Velmi jednoduchá aplikace File fisher[12], která umožňuje organizovat a přesouvat soubory na základě jejich typu. V aplikaci je možno vybrat zdrojovou a cílovou složku. Hlavní funkčností je pak výběr druhů souborů, které se budou přesouvat. Je možno vybrat pouze určité formáty souboru, jako například JPEG, či skupinu formátů, například reprezentující fotografie. Tyto soubory jsou poté na základě požadavků přesunuty či zkopírovány.

**AmoK Exif sorter**

Amok Exif sorter[13] je přenositelnou aplikací spustitelnou na platformách Windows, Linux a Mac.

Ačkoliv aplikace podporuje pouze organizaci fotografií, obsahuje několik velice zajímavých funkcí. Samozřejmostí je možnost přejmenování a přesunu či kopírování souborů do složek, a to na základě jména a metadat fotografických souborů. Navíc program dokáže zobrazit náhled aktuálně zpracovávaného souboru včetně jeho metadat. Zabudovaná je podpora jak metadat ve formátu Exif, tak ve formátu IPTC. Aplikace umožňuje vytvářet profily pro jednotlivé uživatele či zdroje fotografií. Z hlediska uživatelské rozhraní neposkytuje aplikace příliš komfortu díky staršímu grafickému rozhraní.

**Mendeley**

Aplikace Mendeley[14] je účinný nástroj pro organizaci souborů ve formátu PDF, který je tvořen jako správce bibliografické knihovny pro akademické práce. Ze zdrojové složky dokáže na základě obsahu, autora a názvu práce soubory organizovat do adresářové struktury. Soubory, které má program organizovat, je možné vybrat manuálně. Druhou možností je nastavení hlídání určité složky, kterou program v určitém časovém intervalu vždy zorganizuje.

[10] www.lifehacker.com
[11] www.launchpad.net/miranda
[12] www.virtualsoft.weebly.com
[13] www.amok.am
[14] www.mendeley.com

Program dokáže automaticky vytvářet citace jednotlivých děl přímo pro programy jako MS Word, LibreOffice či BibTeX. Zároveň slouží jako prohlížeč PDF souborů, které dokáže otvírat v jednotlivých panelech, mezi kterými lze jednoduše přecházet. Navíc je možné do jednotlivých prací zapisovat své vlastní poznámky. Svou knihovnu poté můžete sdílet s ostatními uživateli na sociální síti, která je součástí aplikace.

Mendeley je možné používat na všech dostupných platformách. Uživatelské rozhraní programu je moderní, přehledné a snadno použitelné.

## 2.3 Metody klasifikace souborů

Jak prudce stoupá množství ukládaných dokumentů, narůstá i potřeba jednotlivé dokumenty zpracovávat. Jedním z procesů správy dokumentů je i jejich klasifikace. Jedná se o druh problému strojového učení, který řeší problém rozdělení dokumentů do tříd či kategorií.

### Definice pojmu klasifikace

Rozdělení objektů do konečného počtu tříd je prováděno na základě jejich atributů. Základem je tréninková množina objektů, pro které je známa výsledná třída. Z toho vyplývá, že se jedná o učení s učitelem. V opačném případě, kdy není známa množina objektů a jejich výsledných tříd, se jedná o učení bez učitele, pak mluvíme o shlukové analýze. Jako příklad nejjednodušší klasifikace se často uvádí rozdělení emailů do kategorií spam a ne-spam. Algoritmus, který klasifikaci provádí, se pak nazývá klasifikátor, což často značí také matematickou funkci. Samotná klasifikace probíhá ve dvou krocích, kdy nejprve probíhá učení, tedy tvorba klasifikačního modelu. Ten musí být schopný klasifikovat tréninkové objekty, u nichž známe výslednou třídu. Druhým krokem je pak samotná klasifikace nových objektů, tedy přiřazení třídy [11].

### Vybrané metody klasifikace souborů

Mezi nejrozšířenější metody klasifikace patří rozhodovací pravidla či rozhodovací stromy a jejich rozšíření v podobě algoritmů ID3 či C4.5. Další známé a často používané metody klasifikace dokumentů jsou Naivní Bayesův klasifikátor, Neuronové sítě, Support vector machines či Algoritmus k-nejbližších sousedů. Vybrané metody jsou podrobněji vysvětleny. Nejdůležitější vlastností klasifikačních modelů je jejich přesnost, která vyjadřuje schopnost modelu klasifikovat neznámá data. Mezi další důležité vlastnosti klasifikačních modelů patří také výpočetní složitost, odolnost vůči chybám či složitost interpretace [11].

#### Rozhodovací pravidla

Jednou z nejzákladnějších metod klasifikace jsou rozhodovací pravidla. Používají se stejně jako rozhodovací stromy, přičemž jejich syntaxe je

$$IF\ Antecedent\ THEN\ Class$$

kde Antecedent je kombinace vstupních atributů a Class je výsledná třída, do které byl objekt klasifikován [2]. Rozhodovací pravidla lze vytvořit přímo z rozhodovacího stromu, a to jako logický součin všech podmínek testů atributů od kořene stromu k listovému uzlu.

Samotná klasifikace probíhá tak, že se postupně vyhodnocují jednotlivé pravidla na klasifikovaném objektu. Nezáleží přitom na pořadí pravidel, jedná-li se o obecný model systému. V případě využití algoritmu C4.5 může být každé pravidlo ohodnoceno vahou reprezentující prioritu daného pravidla[11].

### Rozhodovací stromy

Reprezentování informací v podobě rozhodovacího stromu je známo z mnoha oblastí. Jedním z nejklasičtějších použití je například určování rostlin v biologii. Hlavní výhody jsou především jednoduchost a efektivnost. Limitující vlastnost je především fakt, že rozhodovací stromy nelze využít pro spojitá data. Ta musí být nejprve převedena na diskrétní data, přičemž i výsledek je diskrétní. Rozhodovací strom je definován jako graf stromové struktury, kde vnitřní uzly obsahují testování určitého atributu, načež se na základě jeho výsledku dále větví. Koncové uzly neboli listové uzly, pak reprezentují výslednou kategorii či třídu. Nejprve je třeba vytvořit samotný strom, jenž může být vytvořen pomocí algoritmu. Při tvorbě rozhodovacího stromu se postupuje metodou rozděl a panuj, kdy se tréninkové data rozdělují na menší podmnožiny, než jsou třídy tvořeny jen instancemi téže třídy, přičemž na počátku jsou všechny instance součástí jediné třídy. Tento algoritmus se nazývá *top down induction of decision tree (TDIDT)*, tedy se jedná o postup shora dolů [2]. Požadavkem je nalézt konzistentní strom s tréninkovými daty. Popis algoritmu je v tabulce 2.1.

1. Zvol jeden atribut jako kořen dílčího stromu,
2. Rozděl data v tomto uzlu na podmnožiny podle hodnot zvoleného atributu a přidej uzel pro každou podmnožinu
3. Existuje-li uzel, pro který nepatří všechna data do téže třídy, pro tento uzel opakuj postup od bodu 1, jinak skonči

Tabulka 2.1: Algoritmus TDIDT [2].

Při vytváření stromu často dochází k nežádoucímu jevu, a to k vzniku větví, které mají jen malý význam v rámci rozhodování. Vznikají především vlivem datového šumu. Postupu, kdy se brání vytvoření těchto větví, se říká prořezávání rozhodovacího stromu. Tento proces může probíhat při vytváření (poté mluvíme o prepruningu[2]) nebo po vytvoření celého stromu (postpruning[2]). Výběr kořenového atributu, jakožto i výběr dalších kořenových atributů je nutné nepodcenit. Požadavkem je získat takovou vlastnost, která co nejvíce odliší objekty jednotlivých tříd. Možností, jak takový atribut určit je několik. Mezi nejčastěji využívané patří entropie, informační zisk apod.

**Algoritmus ID3** [11] Tento algoritmus v podstatě pracuje tak, jak je popsán algoritmus TDIDT, tedy také postupuje shora dolů a v každém uzlu volí atribut, který bude použit pro větvení stromu. Důležitou vlastností je využití informačního zisku pro zjištění atributu, který bude strom větvit. Vybírá se nejvyšší hodnota informačního zisku. Jeho rozšířením je pak systém C4.5.

**Algoritmus C4.5** [11] Jedná se o modifikaci ID3. Oproti předchozímu algoritmu umožňuje C4.5 zpracovávat i spojité vstupní data. Podle tréninkových dat jsou spojitá hodnoty atributů rozděleny do intervalů. Také se lépe vypořádá s chybějícími daty. Při tvorbě stromu jsou ignorována, takže nejsou započítávána do výpočtu informačního zisku. Při klasifikaci jsou pak tyto hodnoty predikovány na základě ostatních hodnot stejného atributu.

### Bayesovská klasifikace

Metoda Bayesovské klasifikace je statistickou metodou, která vyjadřuje pravděpodobnost, že byla data správné klasifikována[2]. Metoda vychází z důsledků Bayesovy věty o podmíněných pravděpodobnostech, kterou poprvé zveřejnil Thomas Bayes (1701-1761). Předpokladem je reprezentace dat ve formě vektoru, kdy každá složka vektoru značí hodnotu atributu. Pomocí Bayesovské klasifikace se pak určí pravděpodobnost, že daná instance patří do různých tříd. Výsledkem klasifikace je třída, pro kterou je pravděpodobnost nejvyšší. Bayesův vztah pro výpočet podmíněné pravděpodobnosti má tvar

$$P(c_i|x) = \frac{P(x|c_i) * P(c_i)}{P(x)}$$

Tento vzorec je nazýván Bayesovsým teorémem [1] a má klíčovou roli při třídění pomocí Bayesovské klasifikace, kde $x$ představuje vektor instance dat, kterou aktuálně klasifikujeme, a $c_i$ reprezentuje třídu, do které klasifikujeme. Máme-li tedy například množinu tříd C, která má m prvků, budeme klasifikovat pro třídy $c_1$ až $c_m$. Potom $P(c_i|x)$ je podmíněná pravděpodobnost, že instance patří do do třídy $c_i$.

**Naivní bayesův klasifikátor** Naivní Bayesův klasifikátor je založen na předpokladu, že jednotlivé atributy jsou na sobě nezávislé. Tento model je poměrně jednoduchý, přesto má často velmi dobré výsledky [2]. Pomocí této metody je tak přiřazena třída záznamu na základě nejvyšší aposteriorní pravděpodobnosti [6]. Cílem je tedy najít $c_i$ takové, aby byla aposteriorní pravděpodobnost $P(c_i|x)$ co nejvyšší [1]. Lze využít následující vzorec:

$$c_{max} = \arg\max_{c_i \in C} P(c_i|x) = \arg\max_{c_i \in C} P(c_i) \prod_{k=1}^{n} (t_k|c_i)$$

kde $t_k$ je položka vektoru $x$, jež má právě $n$ položek.

### Algoritmus k-nejbližších sousedů

Metoda k-nejbližších sousedů, často značená jako kNN (k-nearest neighbors), využívá analogie klasifikovaných a tréninkových dat. Mají-li dvě instance něco společného, je pravděpodobné, že budou patřit do stejné třídy. Princip metody spočívá v nalezení nejbližších sousedů dané instance dle atributů. Sousední instance přitom jsou ty, které již existují jako naučené a správně oklasifikované prvky. Poté se provede výběr několika sousedů, přičemž nejčastěji se vyskytující třída u těchto instancí bude výslednou třídou i právě klasifikovaného prvku. Instance je reprezentována vektorem svých atributů. Stěžejní otázkou této metody je vybrání správného k, které určuje počet sousedů. Druhou důležitou části je vybrání správné metriky pro určení podobnosti instancí. Často je využívána eukleidovská vzdálenost, která je definována takto [11]:

$$d(X,Y) = \sqrt{\sum_{i=1}^{n} (x_i - y_i)^2}$$

Mezi hlavní výhody této metody patří jednoduchost návrhu a implementace. Negativy metody pak je vysoká paměťová náročnost, pomalé rozhodování a závislost na zvolené metrice [5]. Oproti porovnání s jediným sousedem, jež často může způsobit chybu, je metoda robustnější [1].

### Neuronové sítě

Metoda klasifikace pomocí neuronových sítí je inspirována lidským mozkem, který obsahuje přibližně $10^{11}$ neuronů[6]. Jedná se o buňky, jež jsou schopny přenášet a zpracovávat informace, současně jsou mezi sebou propojeny pomocí dendritů. Funkce neuronu lze popsat takto: pomocí dendritů neurony zachycují signály. Tyto signály se šíří dovnitř buňky, kde vzniká potenciál. Je-li tento potenciál dostatečně velký, neuron je schopen sám vyslat signál dál [6]. Rychlost přenosu v mozku je až $10^{-3}$ sekundy, zatímco v počítačových systémech se dosahuje rychlostí až $10^{-10}$ vteřiny. Výhodou mozku je ovšem mohutný paralelismus [6].

Obdobně fungují i umělé neurony. Ty mají několik vstupů, na které jsou přivedeny výstupy ostatních neuronů či samotná vstupní data. Vstupy jsou navíc ovlivněny jejich váhou. V samotném neuronu je pak hodnota prahu, která se přičítá ke vstupům. Výstupem je pak výsledek aktivační funkce daného neuronu. Nejjednodušší neuron tak má pouze binární výstup, což umožňuje klasifikovat pouze do dvou tříd. Ovšem jejich spojováním do rozsáhlých sítí se dá docílit klasifikace do libovolného počtu tříd.

Nejrozšířenější způsob propojení jsou tzv. vícevrstvé sítě, které vždy obsahují vstupní vrstvu, výstupní vrstvu a minimálně jednu vnitřní vrstvu. Mezi dvěma vrstvami se pak vždy nachází úplné propojení neuronů, kdy je každý neuron v jedné vrstvě spojen se všemi neurony druhé vrstvy [14]. Nejprve probíhá učení sítě, přičemž mezi nejpoužívanější metody učení neuronových sítí patří metoda zpětné propagace (backpropagation [2]). Použije se tréninková množina, jež je přivedena na vstupní vrstvu neuronové sítě, odkud se dále v síti šíří informace mezi neurony až do výstupní vrstvy, čímž se získá odezva sítě na daná vstupní data. Výsledek klasifikace je poté porovnán s očekávaným výsledkem. V případě, kdy je dosaženo správného výsledku, jsou vazby mezi neurony vedoucí k výsledku posíleny. V opačném případě, kdy naopak není správného výsledku dosaženo, jsou tyto vazby oslabeny [11].

# Kapitola 3

# Návrh aplikace pro organizaci dokumentů

Kapitola se podrobně věnuje návrhu implementace a testování. Nejdříve je rozebrán objektově orientovaný návrhu aplikace s využitím zásuvných modulů. Další část se pak věnuje právě zásuvným modulům, které zajišťují funkčnost pro jednotlivé typy formátů zpracovávaných dokumentů. Poslední část této kapitoly obsahuje návrh uživatelského rozhraní s důrazem na jednoduchost a maximální použitelnost.

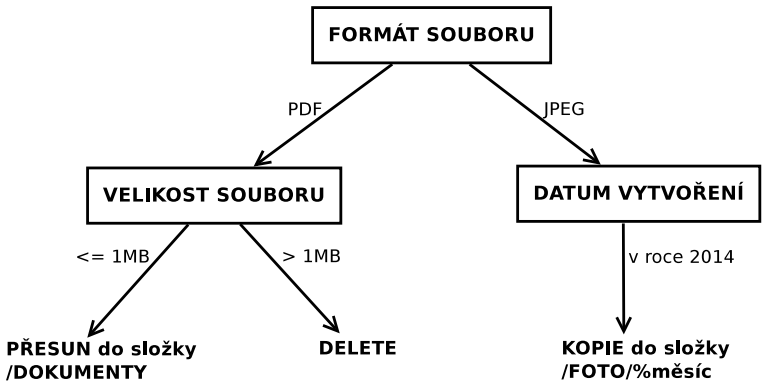
## 3.1 Analýza požadavků

Samotná aplikace lze rozdělit na dvě části. První, základní částí je samotné jádro. Druhou částí jsou zásuvné moduly. Jádro aplikace bude mít na starost především zpracování parametrů, procházení zdrojové složky či operace s dokumenty. Nejdůležitější částí je samotná klasifikace dokumentů, ke které jádro bude využívat zásuvných modulů. Každý zásuvný modul bude reprezentovat jeden typ souborů, jako například zásuvné moduly pro formáty PDF, JPEG a podobně. Základní klasifikaci dle obecných vlastností dokumentů bude provádět samo jádro aplikace, zásuvné moduly budou využity pro složitější a specifičtější operace.

### Jádro aplikace

Stěžejní funkcí jádra aplikace je zpracování požadavků uživatele, výběr požadovaných souborů a provedení operací s nimi na základě pokynů zadaných uživatelem. Po spuštění bude uživatel moci zadat své požadavky, které se skládají z několika částí. Prvním je výběr zdrojové složky, ze které se budou dokumenty vybírat. Zde existuje možnost procházet rekurzivně i další podadresáře. Druhou částí jsou pak pravidla pro výběr a zpracování dokumentů. Tato část se skládá ze dvou kroků, kdy uživatel nejprve vybere podmínky, jež určují soubory pro zpracování daným pravidlem. Druhým krokem je pak výběr operací s danou množinou souborů. Pravidel může uživatel vytvořit několik, podmínkou je jen to, aby se navzájem nepřekrývaly, tedy aby dvě pravidla nemohla zpracovávat stejný dokument. Více o vytváření pravidel je uvedeno dále v této kapitole v sekci Uživatelského rozhraní. Poté, co uživatel dokončí vytváření pravidel, jsou aplikací zpracována. Následně proběhne vytvoření rozhodovacího stromu, na jehož základě se provede s daným souborem vybraná operace, popř. se nezpracuje. V kořeni stromu se vždy porovnává typ souboru, načež jsou porovnávány další vlastnosti. Listy stromu jsou pak třídy, v našem případě reprezentovány

operací, která bude nad daným dokumentem provedena. Příklad jednoduchého rozhodovacího stromu lze vidět na obrázku 3.1.




Obrázek 3.1: Příklad jednoduchého rozhodovacího stromu pro organizaci dokumentů

Procházení zdrojového adresáře probíhá postupně soubor po souboru, v případě požadavku i rekurzivně v podadresářích zdrojové složky. K tomu se dá využít multiplatformní knihovna Boost.Filesystem [3]. Každý dokument je zpracován a projde rozhodovacím stromem, načež se provede výsledná operace. Základní vlastnosti dokumentů zvládne zpracovat samotné jádro aplikace. Jedná se především o takové atributy souborů, které jsou totožné pro všechny formáty. Jádro aplikace tedy bude řešit například velikost, datum vytvoření či název dokumentu, který je aktuálně zpracováván. Pro složitější, především pak pro specifičtější vlastnosti dokumentů, bude jádro využívat zásuvných modulů, které jsou blíže popsány v další části této kapitoly. Každá operace nad souborem je aplikací zapsána do souboru s logy, aby měl uživatel jasný přehled o provedených změnách v adresářové struktuře. Kromě souboru s logy budou tyto informace zobrazovány i za běhu aplikace.

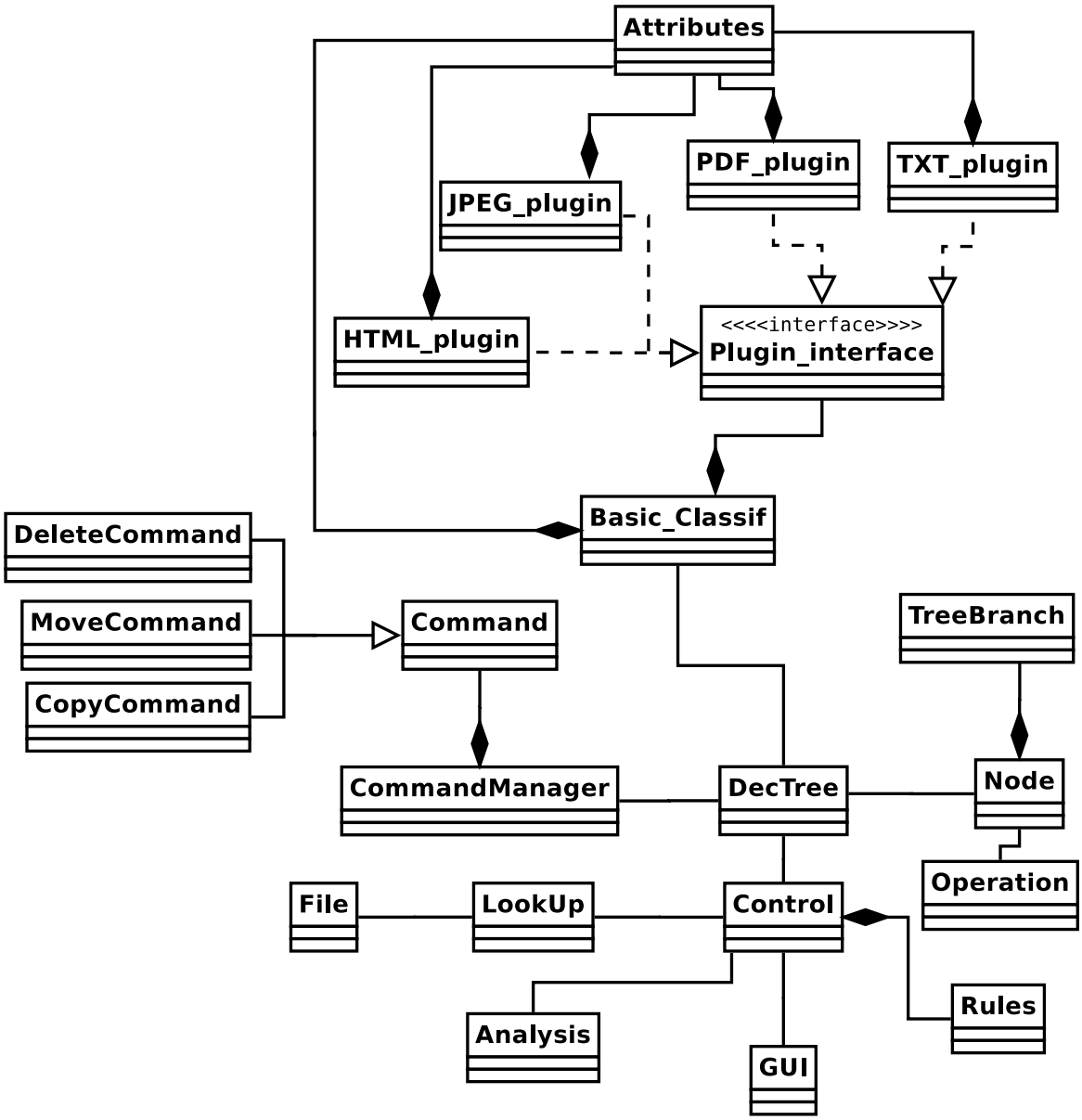
### Zásuvné moduly

Každý zásuvný modul reprezentuje jeden formát dokumentů, přičemž poskytuje možnosti, jak získat specifické informace právě pro daný typ souborů. Odlišné formáty souborů totiž můžou ve své struktuře obsahovat metadata s různým obsahem, jako například Exif metadata u formátu JPEG či ID3 metadata u formátu MP3. Proto je k čtení těchto informací využíváno zásuvných modulů, které zpřístupňují tyto data pro různé formáty dokumentů. Podrobněji se obecnému návrhu zásuvných modulů a podrobnému rozboru jednotlivých modulů věnuje jedna z následujících podkapitol.

## 3.2 Objektově orientovaný návrh aplikace

Vytvořený objektově orientovaný návrh, ač zjednodušený, lze vidět na obrázku 3.2. Základem je třída *Control*, jenž je tvořena podle návrhového vzoru jedináček (Singleton). Tento návrhový vzor [9] specifikuje, jak vytvořit třídu, jež bude mít nejvýše jednu instanci. Jedná se o základní kámen aplikace, který řídí další části. Jednou z těchto částí je grafické uživatelské rozhraní, zde reprezentováno třídou *GUI*, která je zde ovšem výrazně redukována. Ačkoliv samotná třída bude složitější a rozdělena do více tříd, kvůli přehlednosti a jednoduchosti návrhu jsou tyto třídy zde zanedbány. Tato třída bude zajišťovat veškerou komunikaci s uživatelem skrze grafické uživatelské rozhraní, tedy zobrazení hlavního okna

aplikace, okna pro vytváření pravidel a okna pro specifikaci distribuce souborů. Skrze tuto třídu bude třída *Control* zpracovávat požadavky uživatele pro organizaci dokumentů a zajišťovat jejich správné zpracování. Další možností je pomocí třídy *Analysis* provádění analýzy zdrojové složky a zpracování výsledků. Během chodu aplikace, přesněji po dokončení vytváření pravidel, bude vytvořen rozhodovací strom.




Obrázek 3.2: Návrh aplikace — diagram tříd

Rozhodovací strom je reprezentován třídou *DecTree*, jež obsahuje kořenový uzel stromu. Uzly jsou v návrhu zastoupeny třídou *Node*. Každý uzel může obsahovat jak operaci (třída *Operation*), tak i několik dalších větví směřujících k potomkům tohoto uzlu. Větve jsou v návrhu zastoupeny třídou *TreeBranch*. Třída *DecTree* navíc bude zajišťovat i klasifikaci souboru s využitím dalších pomocných tříd.

Jakmile uživatel dokončí vytváření pravidel a spustí samotnou organizaci dokumentů,

nejprve se na základě požadavků uživatele vytvoří samotný rozhodovací strom. Až poté se dostane ke slovu třída *LookUp*. Ta zajišťuje součinnost při práci se soubory, přičemž bude také řešit procházení zdrojové složky soubor po souboru. Tato třída nebude informována o tom, které soubory jsou určené k organizaci, a které nikoliv. To bude řešit až třída obsahující rozhodovací strom. Samotný soubor je reprezentován třídou *File*, který mimo jiné obsahuje i typ souboru, který reprezentuje (např. PDF či JPEG). S objektem této třídy pak začne pracovat i třída *DecTree*.

Právě třída *DecTree* obsahuje samotný rozhodovací strom, přesněji tedy jen kořen stromu. Poté už následuje procházení stromu dle vlastností souboru, které jsou zjišťovány a porovnávány pomocí třídy *Basic_Classif*. Ta může soubor klasifikovat dle základních vlastností, které jsou pro všechny formáty souborů totožné, např. velikost, datum vytvoření či název. V případě, že tato třída nestačí na zjištění požadovaných informací, dostane se k práci jedna ze tříd, které jsou specializované na určité typy souborů (*PDF*, *JPEG*, ..). Ty obsahují specifické metody, které umožňují získat atributy souborů určitého formátu.

Poté, co je dosaženo listového uzlu v rozhodovacím stromu, je provedena požadovaná operace nad daným souborem. Informace o operaci jsou reprezentovány třídou *Operation* a může se jednat o přesun, kopírování či mazání souboru. Po dokončení operace objekt souboru *File* zaniká a pokračuje se v procházení zdrojové složky a práci na dalších souborech.

Samotná operace je provedena třídou *Command*. Jedná se o návrhový vzor Příkaz (Command)[9], kdy z třídy *Command* jsou na základě dědičnosti vytvořeny třídy *MoveCommand*, *CopyCommand* a *DeleteCommand*, přičemž každá z těchto tříd reprezentuje jedinou operaci se souborem. Vykonavatelem je pak třída *CommandManager*, která si uchovává historii provedených operací, díky čemuž je možno realizovat operaci Undo.

## 3.3 Návrh zásuvných modulů

Zásuvný modul zpřístupňuje jádru aplikace možnosti, jak zjistit o souborech různých typů specifické informace. Pro každý formát souboru je právě jeden zásuvný modul. K základní aplikaci náleží zásuvné moduly pro zpracování právě těchto formátů vstupních dokumentů: PDF, JPEG, HTML, TXT. V následující části jsou jednotlivé moduly popsány blíže, včetně nastínění metod, které jednotlivé zásuvné moduly mohou aplikaci zpřístupnit informace o analyzovaných souborech.

**PDF**

Portable document format (PDF), jak již název napovídá, je souborový formát původně vyvinutý firmou Adobe pro ukládání a čtení dokumentů na všech platformách. Obsahem dokumentu nemusí být pouze text, ale i multimédia [8]. Kromě samotného obsahu dokumentu jsou součástí souboru i základní metadata, jež obsahují tyto informace: Autor, Název, Předmět, Klíčová slova.

Zásuvný modul umožní pomocí metod přistoupit k těmto informacím a na jejich základě soubor analyzovat. Nevýhodou ovšem je fakt, že často dokumenty v těchto metadatech neobsahují žádné informace. I proto zásuvný modul pro formát PDF zajistí i možnost analyzovat samotný obsah. Základem bude převod souboru do čistého textu, čímž se ztrácí možnost zpracovat obrázky. S tímto textem je pak možné provést další akce podobné jako v případě souboru TXT, o kterém je více napsáno níže.

**JPEG**

JPEG (Joint Photographic Experts Group) označuje organizaci tvořící standard, me-

todu komprese a samotný souborový formát. Jedná se o sadu ztrátových metod komprese, kdy je možné vytvořit velmi malé komprimované předlohy špatné kvality, či velmi kvalitní komprimované předlohy, což záleží na požadavcích uživatele [7]. Součástí tohoto souboru jsou metadata ve formátu EXIF [13], které jsou do souboru vkládány přímo digitálními fotoaparáty. Mezi ty, které mohou uživatele nejvíce zajímat, patří například tyto informace: Použitý fotoaparát, Datum pořízení, Rozlišení fotografie, Expozice, Clona, Použití blesku.

Cílem tedy je využít pro organizaci souborů právě tyto metadata. Navrhovaný zásuvný modul pro formát JPEG bude obsahovat metody, které budou analyzovat tyto informace. Kromě těchto metadat má největší informační hodnotu samotná fotografie, tedy to, co uživatel vidí na obrázku. Analýza obrazů je podstatně složitější odvětví, nicméně lze využít již existující volně šiřitelné knihovny, které zpřístupňují specifické funkce. Díky tomu lze například zjistit, zdali se na fotografii nachází obličej. Další metody tohoto zásuvného modulu pak budou pracovat s histogramem barev. Na jeho základě lze například získat fotografie, které jsou příliš světlé, či naopak příliš tmavé. Cílem je výběr takových obrázků, které je třeba upravit, aby byly použitelné či úplně smazat.

**HTML**

Značkovací jazyk HTML slouží především pro tvorbu webových stránek. Existuje v mnoha verzích, přičemž aktuálně nejnovější je verze HTML5. Krom toho existuje i rozšíření XHTML. Analyzovat HTML dokumenty lze z mnoha úhlů. Zásuvný modul pro zpracování těchto souborů se zaměřuje především na hlavičkovou část, ve které se nejčastěji uvádí metadata. Mezi nejpoužívanější patří právě tyto: Název stránky, Autor, Popis stránky, Informace pro roboty, Znaková sada, Klíčová slova.

Zásuvný modul umožní pomocí metod analyzovat dokument podle těchto informací. Kromě toho lze zjistit i verzi dokumentu, jelikož ta by měla být zapsána na prvním řádku souboru. Navíc lze, podobně jako u souborů TXT, analyzovat i samotnou textovou část, a to na výskyt určitých slov či délku souboru apod. Další možností je pak zjištění, zdali je HTML dokument validní podle normy specifikované na prvním řádku souboru.

**TXT**

Textový soubor obsahuje jednotlivé bajty, které reprezentují znaky. Tyto znaky lze rozdělit na tisknutelné a bílé znaky (mezera, tabulátor, apod.). Z toho vyplývá, že analyzovat soubory ve formátu TXT lze především dle jejich obsahu.

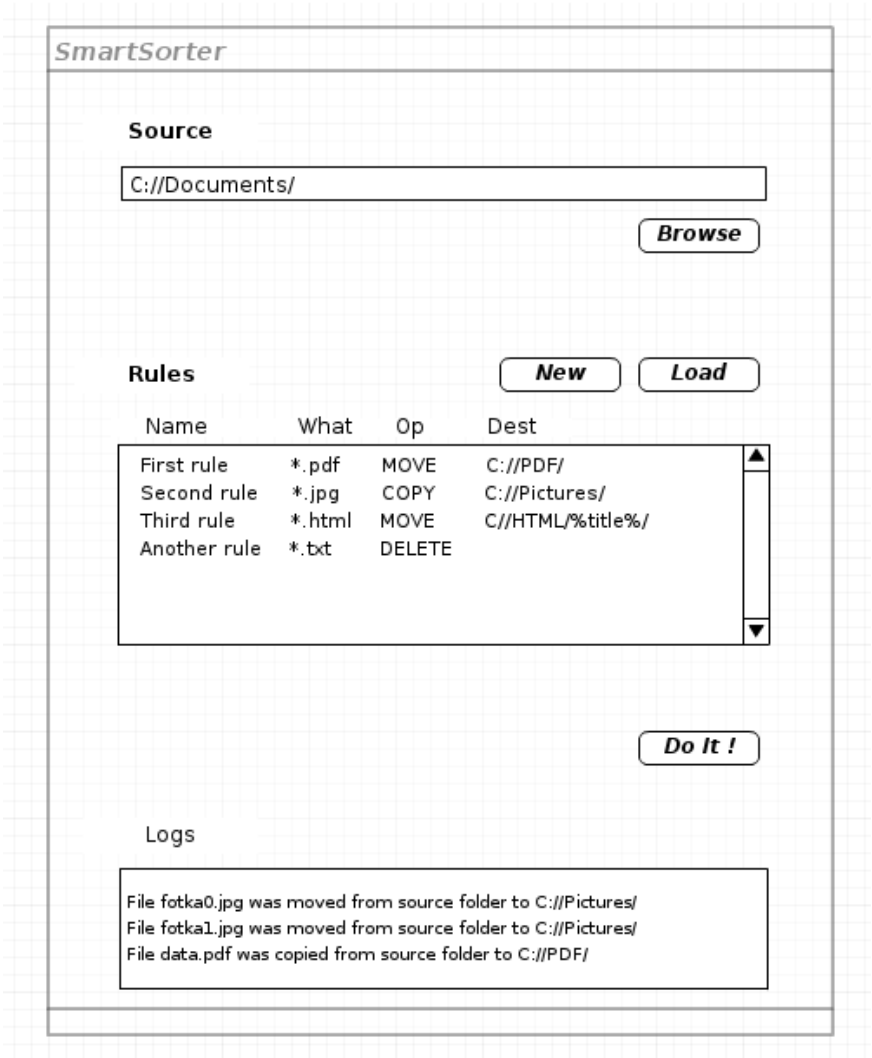
Zásuvný modul pro zpracování těchto dokumentů umožní označit soubor, obsahuje-li hledané slovo či frázi, popř. obsahuje-li dané slovo v určitém množství. Mimoto zjistí statistické údaje o souboru, a to především počet řádků, slov a znaků. Dle těchto informací lze pak soubory organizovat do adresářové struktury.

## 3.4 Grafické uživatelské rozhraní

Hlavním cílem uživatelského rozhraní je umožnit uživateli snadnou orientaci a zpřístupnit funkčnost programu. Aplikace se skládá ze dvou základních oken. V prvním okně je základní přehled o stavu uživatelských požadavků pro organizaci dokumentů. Druhé okno pak zajišťuje přidávání nových pravidel pro organizaci souborů.

### Základní okno aplikace

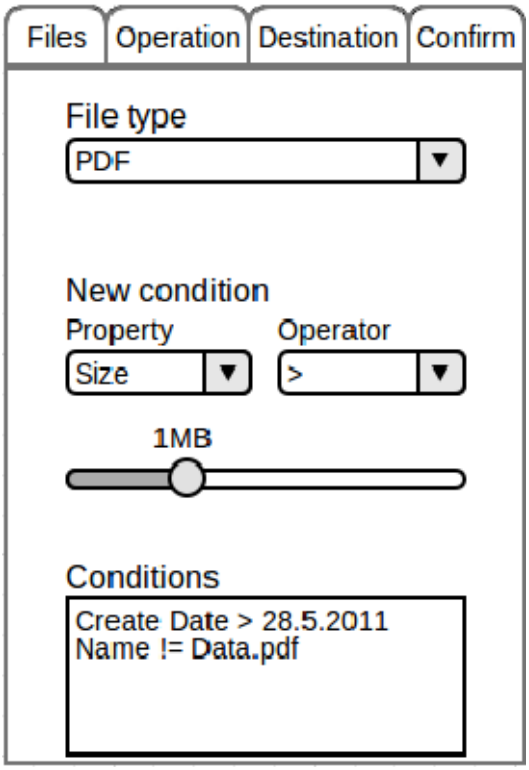
Prvotní náčrt základního okna aplikace lze vidět na obrázku 3.3. Hlavní okno je rozděleno do tří sekcí. První část obsahuje výběr zdrojové složky, která bude zorganizována. Bude tedy obsahovat nástroj pro výběr složky včetně tlačítka pro jednoduchý výběr. Druhá sekce se věnuje pravidlům pro organizaci, kdy lze přidávat a mazat jednotlivá pravidla. Současně budou zobrazena již vytvořená pravidla se stručným popisem, které lze editovat. Pro výběr nového pravidla lze využít již uživatelem uložené pravidla, popř. vybrat z několika předpřipravených pravidel. Pod výpisem pravidel bude i tlačítko pro spuštění práce aplikace. Poslední částí základního okna aplikace je výpis logů. Zde má uživatel možnost zjistit, co vlastně aplikace provedla, popř. zde může být výpis chyb či varování. Stejné informace bude aplikace zapisovat i do souboru s logy.




Obrázek 3.3: Náčrt GUI aplikace — hlavní okno

### Vytváření pravidel

Vytváření pravidel bude probíhat ve více krocích, přičemž náčrt vzhledu pro jeden z těchto kroků lze vidět na obrázku 3.4. První části vytváření pravidel pro organizaci dokumentů je výběr vlastností, které musí soubory splňovat, aby se právě tato operace nad nimi provedla. To spočívá ve výběru formátu souboru, jako například PDF či JPEG. Dále může být tento výběr omezen dle dalších vlastností souboru, proto je v tomto kroku vytváření pravidel možno tyto specifika přidat. V dalším kroku je nutné vybrat operaci, která se bude se soubory splňující podmínky provádět. Na výběr je kopírování, mazání či přesun. Třetí krok se vyskytuje pouze v případě, že se budou soubory přesouvat či kopírovat, jelikož právě v tomto kroku se vybírá jejich cílová destinace. Lze vybrat již existující složku, vytvořit novou či na základě zkráceného zápisu rozdělit soubory dle jejich vlastností do více složek. Jak je vidět na obrázku 3.1, lze tak například fotografie rozdělit do složek na základě měsíce, ve kterém byly pořízeny. V poslední části je pak přehled vytvářeného pravidla, možnost uložit si pravidlo mezi oblíbené a samozřejmě přidat pravidlo do právě prováděné organizace dokumentů.




Obrázek 3.4: Náčrt GUI aplikace — vytváření pravidel

# Kapitola 4

# Implementace a testování

Tato kapitola je rozdělena do dvou částí. Nejdříve je přiblížena samotná implementace aplikace a zásuvných modulů, použité nástroje a knihovny. Také je zde uveden jednoduchý popis tvorby nových zásuvných modulů. V další části kapitoly je pak popsáno provedené testování aplikace, včetně několika případových studií. Aplikace byla vytvořena podle objektově orientovaného návrhu v předchozí kapitole v programovacím jazyce C++ ve standardu C++11[12].

## 4.1 Použité knihovny a nástroje

Vývoj aplikace probíhal v prostředí Linux, distribuce Ubuntu 12.04 LTS. Jako vývojové prostředí byl využit nástroj Qt Creator, k překladu souborů byl využit nástroj GCC ve verzi 4.7.3. Při implementaci byly využity knihovny LibExif, Poppler, dále pak skripty FaceDetect a fdupe a framework Qt.

**Qt** Qt[1] je multiplatformní framework pro vývoj aplikací s grafickým uživatelským rozhraním. Aktuálně je framework šířen jako Open source. Qt nabízí rozsáhlé možnosti pro vývoj uživatelského prostředí, ale také pro práci se soubory či práci s vlákny. Nejnovější verze frameworku je 5.4, k implementaci aplikace byla využita verze 4.8.

**Fdupe** Fdupe[2] je jednoduchý skript v programovacím jazyce Perl pro hledání duplicitních souborů. Skript je spustitelný jak na operačních systémech Windows, tak na Unixových systémech, stačí pouze interpret jazyka Perl. Lze jej využít pod licencí Attribution-NonCommercial Creative Commons, tedy pro nekomerční účely s uvedením autora skriptu.

**FaceDetect** Facedetect[3] je nástroj pro detekci obličeje ve fotografii. Skript je napsán v programovacím jazyce Python, přičemž ke své práci potřebuje i knihovnu OpenCV pro práci s obrazem a videem. Šířen je pod licencí GPLv2.

**LibExif** LibExif [4] je knihovna pro práci s metadaty ve formátu EXIF u souborů ve formátu JPEG. Dokáže metadata jak číst, tak i zapisovat. Knihovna je implementována

[1] http://www.qt.io
[2] http://neaptide.org/projects/fdupe/
[3] http://www.thregr.org/~wavexx/software/facedetect/
[4] http://libexif.sourceforge.net/

čistě v jazyce C a je multiplatformní, aktuálně nejnovější je verze 0.6.21. Šířena je pod licencí GNU LGPLv2.

**Poppler** Poppler [5] je knihovna pro renderování PDF dokumentů. Nejnovější stabilní verze je 0.32. Knihovna je šířena pod licencí GPLv2.

## 4.2 Implementace jádra aplikace

Srdcem aplikace je třída `Control`, která je implementována jako návrhový vzor Jedináček (Singleton [9]). Ta komunikuje s třídami zajišťujícími interakci s uživatelem a sbírá jeho požadavky. Vytvářené pravidla jsou ukládána jako vektor objektů `Rules`. Jakmile uživatel dokončí vytváření pravidel a stiskne tlačítko Start, začne se z uložených pravidel vytvářet rozhodovací strom.

K tomu slouží metoda `BuildTree()` v třídě `DecTree`. Ta postupně projde každé uložené pravidlo a zakomponuje jej do vznikajícího stromu. Každé pravidlo má svou prioritu danou pořadím, jež bylo zadáno uživatelem. Po vytvoření stromu se inicializuje instance třídy `LookUp`, která pomocí metody `GetNextFile()` vrací referenci na další instanci třídy `File`, jež reprezentuje právě jeden soubor ze zdrojové složky. Oproti návrhu, kde bylo plánováno k procházení složek využít multiplatformní knihovnu Boost.Filesystem, byla tato část implementována za pomocí třídy `QDirIterator`, která je součástí frameworku Qt. Pomocí metod `hasNext()` a `next()` lze pak projít rekurzivně vybranou složku soubor po souboru.

Reference na třídu reprezentující soubor je dále předána metodě `DoTheMagic()` třídy `DecTree`. Zde se nejprve vyberou z kořenového uzlu stromu ty podstromy, do nichž může soubor patřit. V kořenovém uzlu je vždy porovnání na formát souboru, přičemž se může jednat o přesně specifikovaný formát (např. JPEG), popřípadě skupinu formátů (např. obrázky). V každém takovém podstromu, kterému formát souboru vyhovuje, se poté hledá nejlepší výsledek pomocí metody `FindBestResult()`, která rekurzivně projde celý podstrom a vrátí uzel s nejvyšší prioritou. Pokud výsledný uzel obsahuje referenci na instanci třídy `Operation`, byl soubor úspěšně klasifikován do nějaké třídy. V opačném případě soubor neodpovídá žádnému z vytvořených pravidel.

Metoda `FindBestResult()` projde vždy všechny větve daného uzlu, přičemž kontroluje, zdali může touto větví pokračovat, a to pomocí metody `IsFileFullfillCondition()` třídy `Basic_classif`. Ta vrací pravdivostní hodnotu na základě porovnání atributu souboru s hodnotou v dané větvi rozhodovacího stromu. V případě, že nedokáže danou podmínku vyhodnotit, zavolá stejně pojmenovanou metodu v jednom z odpovídajících zásuvných modulů.

### Provádění operací se soubory

Jakmile je nalezena požadovaná operace reprezentována třídou `Operation`, jež obsahuje typ operace se souborem a cílovou složku, dostane se ke slovu třída `CommandManager`. Samotné provádění operace je implementována podle návrhového vzoru Příkaz[9]. Nejprve se vytvoří instance třídy odpovídající vybrané operaci, například v případě operace přesunu souboru se vytvoří instance `MoveCommand`. Ta se inicializuje hodnotami nutnými pro vykonání příkazu, tedy adresu souboru a adresu cílové složky. Následně je zavolána metoda `ExecuteCommand()` třídy `CommandManager`, jež dostane jako parametr referenci na právě vytvořený příkaz. Tento

[5] http://poppler.freedesktop.org/

příkaz se provede vyvoláním metody `Execute()` a zároveň se uloží do vektoru vykonaných příkazů.

Díky uchovávání provedených příkazů je možné provést operaci Undo. Každá ze tříd reprezentující jednotlivé operace, jako např. `MoveCommand`, totiž obsahují krom metody `Execute()` i metodu `Undo()`. Samotné operace se soubory, jako například přesun či kopírování, jsou prováděny pomocí třídy `QFile`. Tato třída frameworku Qt je implementována jako návrhový vzor Fasáda [4].

### Ukládání pravidel

Uživatel má možnost si své vytvářené pravidla uložit, aby při opětovném spuštění aplikace nemusel totožné pravidla tvořit znovu. Jelikož je vytvořené pravidlo reprezentováno instancí třídy `Rule`, je třeba tuto instanci uložit do souboru, jedná se tedy o serializaci. V tomto případě je objekt ukládán formou souboru ve formátu XML [10].

O to se stará metoda `SaveRule()`, která uloží atributy objektu do souboru XML. K tomu je využito tříd `QFile` a `QTextStream`. K následnému nahrání pravidla slouží metoda `LoadRule()`, jež využívá nástroje QDomDocument, který je součástí frameworku Qt. Ten umožňuje číst XML soubory a získávat z něj požadovaná data. Jakmile je XML soubor aplikací přečten, je zavolána funkce `IsRuleValid()`, jež ověří správnost nahraného pravidla.

## 4.3 Implementace zásuvných modulů

Pro zásuvné moduly je vytvořené rozhraní, jež je popsáno v sekci Vytváření nových modulů. Jako součást aplikace byly vytvořeny čtyři zásuvné moduly, jejichž implementace je blíže popsána v následující části. Nahrávání zásuvných modulů do aplikace je zajištěno pomocí třídy `QPluginLoader` frameworku Qt, která umožňuje za běhu načítat zásuvné moduly. Ty jsou tvořeny dle rozhraní, které je implementováno jako součást aplikace.

**PDF**

Zásuvný modul pro klasifikaci souboru formátu PDF nabízí možnosti pro porovnávání metadat, jež jsou součástí PDF dokumentu. Kromě toho je možné klasifikovat dle obsahu dokumentu či počtu stran.

Knihovna Poppler nabízí metody pro získání požadovaných informací o souboru. Stačí jen vytvořit instanci třídy `Poppler::Document`, do které se nahraje klasifikovaný soubor formátu PDF. Pomocí metody `info()` je možno získat metadata jako autora, předmět či titulek dokumentu. Metoda `numPages()` pak vrací celkový počet stran dokumentu. Metodou `page()` lze získat objekt třídy `Poppler::Page`, jež reprezentuje právě jednu stránku dokumentu. Tato třída nabízí metodu `search()`, která vyhledává požadovaný řetězec na dané straně a vrací pravdivostní hodnotu informující o výsledku tohoto hledání.

**JPEG**

Zásuvný modul pro formát JPEG nabízí možnost klasifikace souborů jednak dle metadat ve formátu EXIF obsažených v souboru, zároveň pak podle obsahu samotné fotografie. K práci s metadaty je využita knihovna LibExif. Nejdříve se vytvoří instance třídy `ExifData` pomocí funkce `exif_data_new_from_file()`. Následně se získá požadovaný záznam pomocí funkce `exif_content_get_entry()`, jež vrací instanci třídy `ExifEntry`. Následně již stačí jen pomocí funkce `exif_entry_get_value()` získat požadované data k porovnání.

Další možností je klasifikovat dle rozměrů fotografie. K jejich získání je využito třídy `QImage`, jež je součástí frameworku Qt. Nejprve se vytvoří instance třídy na základě samotného souboru JPEG. Následně je metodou `size()` získána instance třídy `QSize`, která je také součástí frameworku Qt. Ta obsahuje samotné rozměry fotografie. Poslední implementovanou vlastností souborů formátu JPEG, pomocí níž lze klasifikovat soubory, je přítomnost obličeje na fotografii. K tomu je využíván skript FaceDetect, jež s parametrem -q vrací návratovou hodnotu značící přítomnost obličeje. Ke spuštění skriptu a získání návratové hodnoty je využita třída `QProcess`, jež je součástí Qt.

**HTML**

Zásuvný modul pro formát HTML umožňuje klasifikovat soubory dle titulku a meta informací v hlavičce HTML dokumentu. Extrahování těchto informací ze souboru je zajištěno pomocí nástroje QtWebKit, což je open source nástroj pro renderování webového obsahu, jež je součástí frameworku Qt.

Do instance třídy `QWebView` je nahrán obsah celého HTML dokumentu. Z této třídy je už možno získat titulek webové stránky. Pro získání metadat je nutno použít třídu `QWebFrame`, do které je nahrán obsah hlavního rámce stránky. Tato třída pak nabízí metodu `metaData()`, jež vrací datovou strukturu `QMultiMap` naplněnou metadaty daného HTML dokumentu. Odtud již stačí jen vytáhnout požadované informace.

**TXT**

Nejjednodušší je zásuvný modul pro formát textových souborů TXT. Ten umožňuje klasifikovat soubory podle statistických údajů souboru jako počet řádků či počet slov. Tyto údaje jsou získávány tak, že se projde celý soubor a jednotlivé řádky či slova se spočítají.

Další možností je klasifikovat dle obsahu dokumentu. Zde se využívá třídy `QString`, do níž se nahraje obsah celého dokumentu. Metoda `contains()` pak vrátí pravdivostní hodnotu značící přítomnost hledaného řetězce.

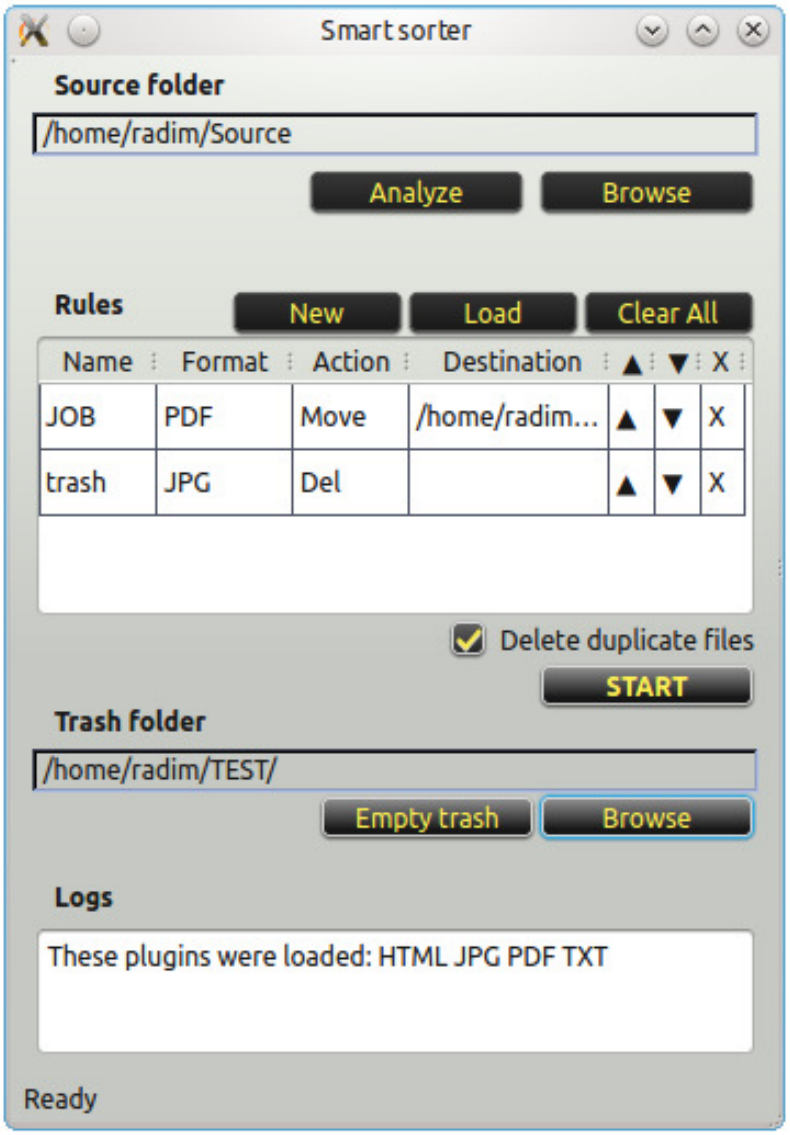
### Vytváření nových modulů

Jako součást vytvořené aplikace bylo implementováno i rozhraní pro vytváření dalších zásuvných modulů `plugin_interface.h`. To obsahuje pouze virtuální metody, které je nutné implementovat v nově vytvářených zásuvných modulech. Jednou z důležitých je metoda `Initialize()`, jež naplní vektor atributů `Attribute`, které je možno u souboru daného formátu použít při klasifikaci. Tyto informace pak využívají metody jako `GetProperities()` či `GetPropertyOperations()` a další, které předávají informace o tom, jaké atributy s jakými operacemi lze použít.

Pro samotnou klasifikaci souboru je pak nezbytná metoda `IsFileFullfillCondition()`. Ta dostane jako parametr samotný soubor, řetězec označující vlastnost k porovnání a uzel rozhodovacího stromu obsahující uživatelem zadaná data. Metoda vyhodnotí, zdali soubor podmínce vyhovuje či nikoliv a vrátí pravdivostní hodnotu.

## 4.4 Grafické uživatelské rozhraní

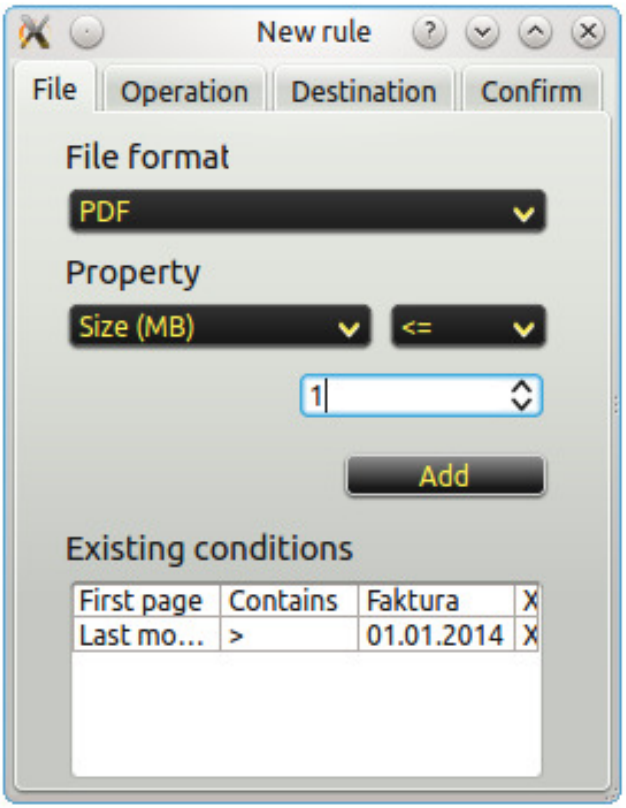
Uživatelské rozhraní aplikace bylo vytvořeno dle návrhu pomocí aplikace Qt Creator, prohlédnout si jej můžete na obrázku 4.1. Jako inspirace pro intuitivní a přehledné GUI po-




Obrázek 4.1: GUI aplikace — hlavní okno aplikace

sloužila aplikace Hazel zmíněná v první kapitole.

Jelikož se jednotlivá vytvářená pravidla mohou překrývat, tedy vybraný soubor může odpovídat více pravidlům, může uživatel přímo v aplikaci jednoduše s vytvořenými pravidly manipulovat a měnit tak jejich priority. Oproti návrhu byla do GUI zapracována i možnost pro výběr složky sloužící jako koš. Do této složky budou přesunuty veškeré soubory označené jako duplicitní či ty, nad kterými bude vykonána operace Delete. Koncept koše byl vytvořen zejména proto, aby se zabránilo nechtěnému smazání souborů. Přímo z aplikace je poté koš možno vysypat, což zapříčiní trvalé odstranění souborů.




Obrázek 4.2: GUI aplikace — vytváření nových pravidel

Vytvářením pravidel provede uživatele jednoduchý průvodce, který je zobrazen na obrázku 4.2, který ve čtyřech krocích získá od uživatele veškeré potřebné informace pro vytvoření pravidla. V záložce, která je zobrazena na obrázku 4.2 lze vidět možnosti přidávání nových podmínek pro klasifikaci dokumentů. V další záložce je možno vybrat si operaci, která bude se souborem provedena. Třetí záložka pak umožňuje specifikovat cílovou složku a distribuci souborů. V poslední záložce je pak nutné pravidlo pojmenovat a popřípadě je možné si jej uložit.

## 4.5 Postup instalace

Aplikace je dostupná ke stažení na adrese https://github.com/radimsvacek/SmartSorter, kde jsou k dispozici veškeré zdrojové soubory včetně návodu k instalaci. Aplikaci je možné

používat a šířit pod licencí GNU GPLv2.

Instalace aplikace na operačních systémech Unix probíhá standardní formou. Nejdříve je tedy potřeba stáhnout veškeré potřebné soubory a nainstalovat knihovny LibExif a Poppler, které jsou nutné pro běh zásuvných modulů, a také framework Qt. Poté je možné aplikaci nainstalovat pomocí následujících příkazů provedených ve složce se staženými soubory: `mkdir build` && `cd build`, `../configure.sh`, `make` a `sudo make install`.

## 4.6 Testování aplikace

Již během implementace byly postupně testovány vždy nově vyvíjené části aplikace na malých souborech dat. Díky tomu byly již během vývoje odladěny závažné chyby. Další fází bylo uživatelské testování, kdy si několik uživatelů vyzkoušelo práci s aplikací. Pomocí tohoto způsobu ověření funkčnosti se podařilo odstranit několik chyb v uživatelském rozhraní i drobné chyby v implementaci.

V závěrečné fázi implementace bylo provedeno zátěžové testování aplikace v rámci tří případových studií, jejichž popis a výsledky jsou rozepsány v následující části. Nejdříve je popsán cíl každého testu a zdrojová data k němu využita. Nakonec jsou popsány výsledky testu a základní statistické údaje. Zobrazen je vždy časový údaj, po který aplikace běžela. Soubory byly přesouvány do složek umístěných na stejném pevném disku. Tyto testy byly prováděny na systému Ubuntu 12.04 LTS, na stroji Intel Pentium P6100 s frekvencí 2.00GHz, 4GB paměti a pevným diskem s kapacitou 500 GB a 5400 rpm.

### Případová studie 1

První testování proběhlo na obsáhlém souboru fotografií, jež čítal celkem 1388 fotografií ve formátu JPEG o celkové velikosti přibližně 3.5 GB. Požadavkem bylo tyto fotografie, jež byly pořízeny během léta 2012 na čtyři různé fotoaparáty, roztřídit do složek dle následující struktury

*/CÍLOVÁ SLOŽKA/ROK/MĚSÍC/DEN/FOTOAPARÁT/*

Zdrojem byla jediná složka, ve které byly umístěny všechny fotografie bez jakéhokoliv rozložení do adresářů. Aplikace během tohoto testu využije zásuvný modul pro formát JPEG, jež bud použit na porovnávání všech vlastností, tedy datum pořízení fotografie a použitý fotoaparát.

**Výsledky** Aplikace dokázala úspěšně klasifikovat a roztřídit všechny fotografie ze zdrojové složky dle požadavků. I včetně operace přesunu souborů byla doba trvání běhu aplikace celkem 134 vteřin. Aplikace tedy stihla klasifikovat a roztřídit přibližně deset souborů za sekundu.

Naopak při opakovaných testech se projevila mnohem vyšší časová náročnost při použití atributu přítomnosti obličeje na fotografii, pro který je využit skript FaceDetect představený dříve v této kapitole. Již při počtu 100 fotografií určených k organizaci stoupl čas nutný k organizaci na 454 vteřin, tedy přibližně 4.5 vteřiny na jeden soubor. Navíc se ukázalo, že skript ne vždy správně rozpozná přítomnost obličeje na fotce. Jedním z problémů je velká citlivost na kvalitu fotografie. I s kvalitní fotografií si ovšem neporadí vždy správně, výsledná úspěšnost klasifikace byla pod 50%.

### Případová studie 2

Ve druhé studii byl prověřen zásuvný modul pro formát PDF. Zdrojem byly netříděné soubory v tomto formátu, přesněji 227 souborů o celkové velikosti 240 MB. Cílem tohoto testu bylo rozřadit dokumenty podle několika kritérií. První skupinu souborů tvořily faktury, které bylo potřeba umístit do složky k tomu určené a rozložit je podle roku a měsíce vytvoření. Tyto soubory obsahují vždy na první straně slovo "Faktura". Druhou skupinou souborů byly tzv. katalogové listy (datasheet) k zesilovačům různých výrobců. Pokud první strana dokumentu obsahovala slovo "Amplifier", byl tento soubor vybrán. Navíc byly tyto soubory distribuovány do složek dle přítomnosti názvu několika výrobců. Poslední skupinou souborů pak byly různé dokumenty vydané Státním zemědělským intervenčním fondem. Jeho zkratka SZIF včetně loga byla vždy přítomna na první straně dokumentu. Pokud navíc obsahovaly dokumenty frázi "rychle rostoucí dřeviny", byly přesunuty do podadresáře RRD. Jednotlivé dokumenty byly navíc distribuovány podle roku, kdy proběhla poslední modifikace souboru.

**Výsledky** Vysoká úspěšnost klasifikace byla dosažena i v druhém testu, který trval přibližně 220 vteřin. Celkem bylo klasifikaci podrobeno 436 souborů, z nichž bylo 243 vybráno pro určitou operaci. První skupina dokumentů obsahovala celkem 87 souborů, z nichž 84 bylo úspěšně klasifikováno. Zbylé tři se knihovně Poppler nepodařilo přečíst. Do třetí skupiny pak patří 22 souborů, z nichž bylo úspěšně klasifikováno 19. Absence tří souborů je zapříčiněna přítomnosti obrázkového loga na první straně dokumentu namísto textového vyjádření zkratky "SZIF". Největší problémy vznikly při klasifikaci druhé skupiny dokumentů. Zde se podařilo úspěšně klasifikovat 141 souborů, zatímco 69 souborů se klasifikovat nepodařilo. To bylo ovšem způsobeno faktem, že se jedná o skenované listy papíru. Jedná se tedy o obrázky, nikoliv o text. Naopak úspěšně byly dokumenty roztříděny i podle druhého parametru výrobce součástky.

Při opakování testu s vynecháním podmínky pro vyhledání fráze "rychle rostoucí dřeviny"v celém dokumentu bylo znatelné zrychlení celého procesu klasifikace dokumentů, když celý test trval 107 vteřin, tedy přibližně polovinu. Bylo to zapříčiněno zejména některými mnohastránkovými dokumenty, které bylo nutné projít celé. Naopak při testování bylo zjištěno, že valná většina dokumentů ve formátu PDF nevyužívá metadata, a když už je obsahuje, tak často jen neúplné. Ani dokumenty získané od státní správy metadata neobsahovaly. Přesto je práce s metadaty velmi rychlá, proto by mohla najít své využití.

### Případová studie 3

Poslední test proběhl nad složkou stažených souborů. Ta obsahovala celkem 78 souborů různých formátů o celkové velikosti 8.4 GB, umístěných v samotné složce i několika podadresářích. První požadavkem bylo smazat všechny duplicitní soubory tak, aby zůstal jediný výskyt souboru. Dále bylo požadováno roztřídit soubory dle obecných formátů do složek obsahující Video, Audio a Obrázky. Zbylé soubory starší než rok bylo třeba smazat. Všechny ostatní soubory byly přesunuty do složky Ostatní. Podle zadání bylo tedy vytvořeno pět pravidel, přičemž byla vybrána i možnost mazání duplicitních souborů.

**Výsledky** Všechny soubory aplikace správně klasifikovala a zařadila do správných složek. Tento test tak ověřil schopnost aplikace pracovat se skupinami formátů, jako jsou například obrázky v nejrůznějších formátech (např. JPEG, GIF, PNG apod.). V rámci klasifikace lze

využít i možnost vybírat všechny soubory bez ohledu na formát. Vzhledem k nízkému počtu souborů byla klasifikace velmi rychlá, kdy trvala necelou vteřinu. Současně aplikace přesunula do uživatelem vybrané složky sloužící jako koš pět duplicitních souborů a také soubory starší než rok nezapadající do žádné z vybraných kategorií.

Během prvních pokusů testu bylo zjištěno, že aplikace nedokáže zařadit do skupin souborů více formátů dokumentů. Na základě tohoto poznatku byly tyto skupiny rozšířeny o několik dalších formátů souborů. Výhodou je fakt, že úprava skupin formátů je velmi jednoduchá a může být v budoucnu s příchodem nových formátů souborů lehce upravována.

### Zhodnocení provedených testů

Testováním byla ověřena správná funkčnost aplikace i implementovaných zásuvných modulů. Naopak bylo zjištěno nevyhovující chování skriptu FaceDetect. I proto bylo zahájeno jeho nahrazení, kdy bude využita knihovna Dlib. Statistické údaje o jednotlivých testech jsou v tabulce 4.1.

| Test č. | Počet souborů | Velikost | Klasifikováno souborů | Doba testu |
|---|---|---|---|---|
| 1 | 1388 | 3.5 GB | 1388 | 134s |
| 2 | 436 | 286 MB | 243 | 225s |
| 3 | 78 | 8.4 GB | 73 | 1s |

Tabulka 4.1: Výsledky jednotlivých testů.

# Kapitola 5

# Závěr

Cílem této práce bylo navrhnout a implementovat aplikaci pro inteligentní organizaci dokumentů, která dokáže jednotlivé soubory na základě jejich vlastnosti rozdělit do adresářové struktury. Aplikace byla implementována dle vytvořeného objektově orientovaného návrhu v jazyce C++11. Aplikace dokáže klasifikovat soubory jen podle základních vlastností dokumentů, zatímco vytvořené zásuvné moduly dokáží klasifikovat soubory i podle specifických vlastností jednotlivých typů souborů. Součástí základní aplikace jsou i čtyři zásuvné moduly pro klasifikaci souborů ve formátu PDF, JPEG, HTML a TXT. Pomocí vytvořeného rozhraní je možné jednoduše vytvářet další zásuvné moduly, které aplikaci umožní klasifikovat další typy souborů. Výsledná aplikace je plně funkční a umožňuje jednoduchou cestou uživateli spravovat své dokumenty. Testováním byla ověřena správnost klasifikace dokumentů a provádění operací se soubory.

V rámci pokračování projektu by bylo vhodné rozšířit možnosti klasifikace dokumentů implementací nových zásuvných modulů pro další formáty běžně používaných souborů. Jelikož se použitý nástroj FaceDetect neosvědčil, již bylo zahájeno jeho nahrazení knihovnou Dlib. Další možnou cestou je rozšíření aplikace o práci s archívy, kdy by bylo možné přidat novou operaci pro komprimaci dokumentů, či naopak rozbalování archívu dle určitých pravidel. Vzhledem k automatizaci činnosti by bylo do budoucna vhodné i rozšíření aplikace, které by umožňovalo periodické opakování činnosti na pozadí bez nutnosti zásahu uživatele nad vybranou složkou v uživatelem daném časovém období či sledování vybraných složek s možností automatického spouštění činnosti aplikace.

# Literatura


[1] BARBER, D.: *Bayesian reasoning and machine learning*. Cambridge University Press, 2011, iSBN 978-0-521-51814-7.

[2] BERKA, P.: *Dobývání znalostí z databází*. Academia, 2003, iSBN 80-200-1062-9.

[3] DAWES, B.: Boost Filesystem Library. `http://www.boost.org/doc/libs/1_38_0/libs/filesystem/doc/index.htm`, 2008-08-14 [cit. 2014-12-16].

[4] EZUST, A.; EZUST, P.: The facade pattern. `http://www.ics.com/designpatterns/solutions/facade.html`, 2012-03-02 [cit. 2015-03-22].

[5] HOUDEK, M.; SVOBODA, T.; PROCHÁZKA, T.: Klasifikace podle nejbližších sousedů Nearest Neighbour Classification [k-NN]. `http://cmp.felk.cvut.cz/cmp/courses/recognition/zapis_prednasky/zapis_01/4/rpz4.pdf`, 2001-06-06 [cit. 2014-11-26].

[6] MITCHEL, T. M.: *Machine learning*. McGraw-Hill, 1997, iSBN 0-07-042807-7.

[7] MURRAY, J. D.; VANRYPER, W.: *Encyklopedie grafických formátů*. Computer press, 1997, iSBN 80-7226-033-2.

[8] PADOVA, T.: *Acrobat PDF Bible*. IDG Books, 1999, iSBN 0-7645-3242-1.

[9] PECINOVSKÝ, R.: *Návrhové vzory*. COMPUTER PRESS, 2007, iSBN 978-80-251-1582-4.

[10] QUIN, L.: Extensible Markup Language (XML). `http://www.w3.org/XML/`, 2015-04-20 [cit. 2015-04-22].

[11] RYCHLÝ, M.: Klasifikace a predikce. `http://www.fit.vutbr.cz/~rychly/public/docs/classification-and-prediction/classification-and-prediction.pdf`, podzim 2005 [cit. 2014-11-26].

[12] STROUSTRUP, B.: C++11 - the new ISO C++ standard. `http://www.stroustrup.com/C++11FAQ.html`, 2014-09-05 [cit. 2015-04-12].

[13] TACHIBANAYA, T.: Exif file format. `http://www.media.mit.edu/pia/Research/deepview/exif.html`, 1999-12-19 [cit. 2014-12-12].

[14] VONDRÁK, I.: Neuronové sítě. `http://vondrak.cs.vsb.cz/download/Neuronove_site.pdf`, 2009 [cit. 2015-05-05].

# Příloha A

# Obsah CD

Obsah přiloženého CD má následující strukturu

- README - popis obsahu CD
- xsvace02.pdf - soubor obsahující tuto práci ve formátu PDF
- /doc - zdrojové soubory k sestavení této práce
- /src - zdrojové soubory k aplikaci
- /src/INSTALL - návod k překladu a instalaci aplikace
- /src/README - popis aplikace a možnosti jejího sestavení
- /src/doc - kompletní dokumentace k aplikaci